Why This Book Matters

By Gary Stager

Millions of the miraculous low-cost microcontroller development board, the BBC micro:bit, are in use across the globe. This new book takes a comprehensive approach to teaching children to make and code exciting interactive projects using the micro:bit as its "brain board"—expanding the range of what often happens in classrooms.

The Invent to Learn Guide to the micro:bit combines coding, craft, and creativity to teach physical computing, engineering, computer science, and electronics to students in grades 4-9. Authors Pauline Maas and Peter Heldens designed whimsical projects sure to delight a wide variety of kids while teaching important computing and engineering concepts.

The Invent to Learn Guide to the micro:bit is built upon the intellectual traditions of project-based learning and constructionism. Readers are provided with just enough inspiration and explanation to make projects, and the resulting knowledge, their own. This book can be used by students, but may also be used as a resource for creative teachers looking for project inspiration or as the basis for a course of study.

Projects matter

I may be biased, but *The Invent to Learn Guide to the micro:bit* is the gold standard in books about robotics, programming, and engineering for children. The book is not only distinguished by the ways in which it expands the range of what's possible in classrooms, but also in the approach it takes to teaching S.T.E.M. concepts to children (and their teachers).

> *This book's ingenious projects celebrate the culture of childhood.*

Most projects in *The Invent to Learn Guide to the micro:bit* use Microsoft MakeCode as their programming environment and treat "Make Code" literally. Each project in the book includes a making component and computer coding to make things happen. Every project employs one or more micro:bits. Young inventors are invited to embellish, personalize, add creative touches, and expand the functionality of each project via computer programming, art, and design.

Too many "maker" books suffer by speeding from simple to impossible within a few pages. Others promise robotics or tinkering but deliver arts and crafts. This book is different. The projects increase in complexity gradually and are developmentally appropriate. Every project requires construction and coding. This broadens the appeal for a diverse population of children. Reader-tinkerers are encouraged to tackle increasingly sophisticated projects and personalize them. There are plenty of challenges presented for high-flyers, giving kids a good look at the micro:bits remarkable functionality and their own potential.

Playing with powerful concepts

A common pedagogical error is made by many educators using computers in the classroom who believe it is important to learn all the menu options and features of a software package. As a result, some teachers require students to memorize such technical details, often without ever using the functionality being studied. This is not only a waste of time, but exactly the wrong strategy, especially when using web-based software that is likely to change and be upgraded over time, like MakeCode. The most practical approach to use software like MakeCode is to understand it conceptually. When such fluency develops, you can find the tool or feature you need, regardless of the current interface. Such skills transfer more readily to different coding environments as well.

Occasionally, the book casually introduces new programming concepts with little or no explanation. Subsequent projects utilizing that concept are accompanied by more explanation. This choice represents a deliberate pedagogical stance. The priority should be rich experience over rhetoric. Piaget teaches us that "knowledge is a consequence of experience." Action concretizes the learning process. For example, using a "random" block in the fortune teller project creates an opportunity for "messing about" with the mathematically significant concept of randomness in a natural playful fashion. No long explanation is required.

This book's ingenious projects celebrate the culture of childhood. Critical skills are developed without straining them through the antiseptic lens of vocational training or increasing test scores. Playful does not mean simple or mindless. The book seeks to democratize computer science and physical computing by appealing to a wide variety of students and teachers.

The projects in *The Invent to Learn Guide to the micro:bit* demonstrate powerful computational ideas while requiring problem solving and persistence. In fact, I am unaware of another text about MakeCode programming that introduces more computing concepts to children. Readers will deal with loops, variables, conditionals, sensory inputs, actuator output, timing, synchronization, parallelism, randomness, inequality, procedurality, radio communication, data collection and analysis, coordinates, plus a host of other computer science concepts situated within a project approach.

Students are also invited to explore engineering principles, design, simple electronics, precision, motors, lights, sensors, inputs, friction, mechanical advantage, and a host of other physical science concepts. Rather than being reduced to a list of vocabulary words, these powerful ideas on the frontier of modernity come alive when learners are engaged in playful, meaningful, and creative projects.

Debugging – a 21st century skill

This book is based on a conviction that children and their teachers are competent. They are capable of reproducing block-based computer programs accurately—a contemporary literacy skill. When something does not work with their physical model or computer program, debugging is required. Debugging develops critical habits of mind. Its importance cannot be overstated. Seymour Papert, one of the inventors of Logo, the first programming language for children, said:

> *Many children are held back in their learning because they have a model of learning in which you have either 'got it' or 'got it wrong.' But when you program a computer you almost never get it right the first time. Learning to be a master programmer is learning to become highly skilled at isolating and correcting bugs... The question to ask about the program is not whether it is right or wrong, but if it is fixable. If this way of looking at intellectual products were generalized to how the larger culture thinks about knowledge and its acquisition we might all be less intimidated by our fears of "being wrong" (Papert, 1980).*

A book filled with links to someone else's code may provide the illusion of efficiency but does so at the expense of learning. Creating short block-based programs manually is not a great burden for children. It encourages them to develop fluency in the software environment, read the code, predict its behavior, and become more skilled debuggers when inevitable errors crop up.

Physical computing

Physical computing is the product of making things and programming. The projects in this book move beyond arts and craft projects to add intelligence and interactivity to one's inventions.

Computer programming perfectly matches a young person's remarkable capacity for intensity and yet far too few students learn to program in school. Programming experiences turn the computer into an intellectual laboratory and vehicle for self-expression. Schools have long overvalued learning with one's head while the future requires learning with one's heart, hands, and head equally. Regardless of the path today's student follows, they will be required to solve problems that involve making things out of bits or atoms. The projects in this book require both and are designed to appeal to a wide variety of tastes, styles, and interests.

Small but mighty

The micro:bit is a new species of technology, the microcontroller development board (MDB). Like other microcontrollers such as Arduino, the micro:bit is a little brainboard featuring pins for inputs and outputs. Sensors report information to the embedded microprocessor. The program you write decides what to do with that information and then produces a result by sending instructions to lights, motors, speakers, displays, or other devices. This is achieved by creating circuitry connecting input and output devices to the microcontroller, usually via breadboards or soldering.

A microcontroller development board, such as the micro:bit, has input and output features built right into the board. This lowers the barrier to entry by reducing costs, confusion, and complexity for beginners while offering sufficient power for advanced projects. While you can and will learn a bit about electronics working with the micro:bit, all of the messy bits about circuitry, voltage, resistance, and more have disappeared. Thanks to the micro:bit, novices can have their first physical computing projects built and programmed within minutes.

> *For many reasons, the micro:bit may be the physical computing platform of choice in schools for the next decade.*

Software matters!

For the past decade, the widely popular family of Arduino microcontrollers has been the choice for simple physical computing projects. However, for educators hoping to introduce students to these powerful concepts, the software used for programming the Arduino is a high hurdle—difficult to learn and use. Many young programmers simply repurpose pieces of code found online or give up on the picky syntax and obscure rules.

This is an enormous problem for learners, particularly children. Why should one's imagination be confined to what may be Googled? If you can invent a physical artifact, "brick-by-brick" through tinkering, you should be able to do the same with the program you write to control your creation. There is much to be learned by reading another person's program/code and modifying it, but that should not be the only path to programming or physical computing. Imagine teaching writing by giving students twenty different words and telling them that their storytelling must be limited to only those words.

Today, the micro:bit is taking K-12 education by storm, not just because it is inexpensive, but because it is programmable by anyone. That is because the micro:bit was designed for the education sector and its developers took software seriously. Software is the nervous system of physical computing. Great hardware and the software powering it should be generative. "Messing about" with the micro:bit and the programming environments discussed in this book should generate project ideas. Each success generates other ideas. Bugs invite debugging.

The micro:bit is special

My recent book, *Twenty Things to Do with a Computer – Forward 50*, (Stager, 2021) celebrates the remarkable 1971 paper by Seymour Papert and Cynthia Solomon in which they provide a vision for how children may learn into the future (Papert & Solomon, 1971). More than fifty years ago, Papert and Solomon described the sorts of physical computing projects featured in this book. The variable that made such learning experiences rare for a half century of school children is the cost of hardware.

The micro:bit represents this paradigm shift in access and affordability. It is sufficiently powerful and flexible to meet the needs of learners and create opportunities beyond the limits of our imagination. It may just be that rarest of educational technology unicorn.

The micro:bit ticks several important boxes for educators. It's inexpensive and versatile—approximately $15, with a USB cable, battery box, and batteries. The micro:bit also works with Macs, PCs, Chromebooks, iOS, and Android devices. Paired with MakeCode, a free programming language, it is sufficiently powerful and flexible to meet the needs of learners and create opportunities beyond the limits of our imagination.

For many reasons, the micro:bit may be the physical computing platform of choice in schools for the next decade.

Go for it!

Since the micro:bit is a chip as cheap as chips, educators are more inclined to allow kids to use them in a daring fashion. Use one to learn coding, embed one in a robot, use another to control that robot remotely. Sew a micro:bit into your interactive garment, control a classroom puppet theatre with a pile of them, build a school weather station, or invent something new.

The BBC micro:bit, MakeCode, and this book combine to fuel remarkable ingenuity and powerful ideas for young people. We can't wait to see what teachers and students do with these remarkable resources!

Veteran teacher educator, Gary Stager, Ph.D., is president of Constructing Modern Knowledge Press, the publisher of The Invent to Learn Guide to the micro:bit.

References

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. Basic Books.

Papert, S., & Solomon, C. (1971). *Twenty things to do with a computer* (Artificial Intelligence Memo # 248, Issue.

Stager, G. (Ed.). (2021). *Twenty things to do with a computer forward 50: Future visions of education inspired by Seymour Papert and Cynthia Solomon's seminal work.* Constructing Modern Knowledge Press.